This document is published in:

*Evolving Systems* (2014), 5 (1), 65-72.
DOI: 10.1007/s12530-013-9085-6

# Adaptive Evolving Strategy for Dextrous Robotic Manipulation

**César Arismendi · David Álvarez · Santiago Garrido · Luis Moreno**

**Abstract** A robot task can be represented as a set of trajectories conformed by a sequence of poses. In this way it is possible to teach a mobile robot to accomplish a manipulation task, and also to reproduce it. Nevertheless robot navigation may normally introduce inaccuracies in localization due to natural events as wheel-slides, causing a mismatch between the end-effector and the objects or tools the robot is supposed to interact with. We propose an algorithm for adapting manipulation trajectories for different locations. The adaptation is achieved by optimizing in position, orientation and energy consumption. The approach is built over the basis of Evolution Strategies, and only uses forward kinematics permitting to avoid all the inconveniences that inverse kinematics imply, as well as convergence problems in singular kinematic configurations. Manipulation paths generated with this algorithm can achieve optimal performance, sometimes even improving original path smoothness. Experimental results are presented to verify the algorithm.

**Keywords** Manipulation Planning · Evolution Strategies · Adaptive Systems

## 1 Introduction

The fundamental purpose of robots is to help humans in a variety of difficult tasks, enabling people to increase their capabilities of strength, energy, speed, memory, and to operate in hazardous environments through telerobotics. Service robots, more precisely mobile manipulators, incorporate one or two robotic manipulators and a mobile base, and must accomplish complex manipulations tasks interacting with tools or objects.

In order to accomplish robotic manipulation, inverse kinematics is required to determine the manipulator configurations given an endpoint position and orientation coordinates. Usually, approximated numerical methods are used to solve inverse kinematics due to the complexity and sometimes indeterminacy of analytical solutions. Many numerical methods have been proposed; the most common methods are based on the Newton-Raphson method (Goldenberg et al 1985; Gupta and Kazerounian 1985) and the damped least squares method (Mayorga et al 1992; Chiaverini et al 1991), but these methods have convergence problems in singular kinematic configurations because of their dependence on the Jacobian Matrix. Another approach to solve this problem consist of formulating the inverse kinematics as a constraint optimization problem. This way all singularities are avoided by using forward kinematics instead. In order to solve this optimization problem, several methods have been proposed, from classic methods like conjugate gradient and the Cyclic Coordinate Descent (Wang and Chen 1991), to artificial intelligence methods such as neural networks (Oyama et al 2001; Bao-Liang and Ito 1995; Morris and Mansor 1997; Guez and Ahmad 1988; Tejomurtula and Kak 1999; Thiang and Pangaldus 2009), Fuzzy Logic (Kim et al 1993; Borboni 2001; Yang et al 2001; Kumbla and Jamshidi 1994; Shen et al 2006) and evolutionary algorithms (Tabandeh et al 2006; Parker et al 1989; Huapeng and Handroos 2000). Sampling-based algorithms that rapidly generate solutions have also been widely used, in particular the Rapidly-exploring Random Tree (RRT) has

César Arismendi, David Álvarez, Santiago Garrido and Luis Moreno
Department of Systems Engineering and Automation, Carlos III University of Madrid. Leganés, Madrid, Spain.
e-mails:(carismen, dasanche, sgarrido, moreno)@ing.uc3m.es

been used in a broad range of motion planning scenarios. Bertram et al (2006) propose an algorithm for manipulation planning based on exploring a connected free component of the configuration space with a single RRT. An algorithm based on the same approach is presented in Vande Weghe et al (2007), but rather than randomly extending the search tree out from the node closest to the workspace goal, it computes a goal directed action using the Jacobian Transpose. RRT approaches stand out because of its ability to manage high number of dimensions in a reasonably short time period, but as drawback, paths generated through these approaches lacks smoothness, which could lead to malfunctioning of the robot through abrupt movements.

The Evolution Strategies (ES) have proven to be robust and versatile by not depending on any continuity or derivability condition and have been used successfully in various disciplines. In (González et al 2009) Differential Evolution (DE) is used to find an optimal manipulation path. This algorithm presents graceful properties relative to the approaches mentioned before. First, it needs no determination of the inverse kinematics. Second, all poses are executed in forward kinematics, which implies that all poses are reachable and valid. Third, the generated path tends to be smooth due to the energy consumption optimization. However our work additionally includes mobile base disturbance, self-adjusting parameters and an improved algorithm implementation, that takes far less computation time to execute. This work is an expansion from our previous work Arismendi et al (2012). Now further experimentation and the real robot implementation is presented.

The objective of this work is to adapt or provide previously learned manipulation tasks to other locations in a short period of time. A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed (Laplante 2004). Tasks considered here include reaching tools and objects; we considered these tasks to have a five seconds tolerance term. With this definition, the following algorithm presented here can be categorised as real-time, as long as the tolerance time is not outperformed. Operations completed after the deadline are considered useless.

Given a manipulation task in configuration space described by an end effector path in Cartesian space with a specific position and orientation of the robot base, a new manipulation path is generated in real-time using a modified approach of the basic ES scheme, assuming different initial conditions from the learned path. The ES algorithm implemented here uses criterions as simple as possible, minimizing computational burden, and reaching fast optimal results that coherently describe a manipulation task.

After manipulation path for a defined task is obtained, the challenge is to replicate it. In mobile robotics, localization error is a recurring problem caused by imperfections present on the floor surfaces and in robot hardware, causing wheels to slide among other errors (Xu and Collins 2009); in consequence, robot bases would rarely ever be situated in the exact same position and orientation where tasks were learned. Nonetheless, it is possible to determine the robot's exact position through measure equipment such as lasers and 3D cameras. Once the new location is obtained, the ES algorithm is applied to adapt the learned path of the original accomplished task, to a new sequence of poses.


**Fig. 1** Mobile Robot experimental platform MANFRED-2.

This paper is organized as follows. In Section II, the manipulation path adaptation problem is explained. Section III describes in detail the used ES algorithm with the proposed modifications to the original scheme. Section IV presents the experimental results obtained by testing the algorithm in simulations as well as in the real robot. The mobile manipulator used here is MANFRED-2 (Fig. 1), which is an experimental platform developed by our research group. This robot has

some built-in sensors, which have been used in this work. Finally, conclusions are presented in Section V.

## 2 Path Evolutionary Adaptive Problem

For a mobile manipulator a task may be defined as a sequence of points in the Cartesian space defining a path, (González et al 2009). This sequence of joint configurations is defined as:

$$\Omega_l = \{\mathbf{q}_k\}, \qquad k = 1, 2, \ldots, N, \tag{1}$$

where $\mathbf{q}_k \in \Re^n$ is a vector of joint variables $q_{k,j}$,

$\mathbf{q}_{k,j} = (q_{k,1}, \ldots, q_{k,j}, \ldots, q_{k,n})^T$.

A given free path $\Omega_l$ is assumed to be known. This path describes the robot's goal or task, and may be obtained by learning methods through imitation, teleoperation, teaching techniques or telemetrics systems. After a displacement, the initial location of the robot base is different but not far from $\Omega_l$, and this path must be adapted to the new location to complete the desired task.

By optimizing the end effector's position and orientations errors a new path is obtained for a predefined task. The trajectory is smoothened by considering the energy consumption, and it is determined by the sum of the manipulator joints displacements as

$$\zeta(\Omega) = \frac{1}{2\pi} \sum_{k=1}^{N} |\mathbf{q}_k - \mathbf{q}_{k-1}| \tag{2}$$

In evolving methods, the step length is the disturbance introduced to design variables in order to change and evolve them from initial to optimum positions, this variation parameter is denoted by $\sigma$. In manipulation planning, the end-effector's position can be changed in only one axis with a movement, while orientation axes are hard coupled, varying all or at least two axes simultaneously when rotation over an axis is executed. This makes orientation optimization harder and more computational time consuming than that of position, as small changes in position could generate large variations in orientation. To overcome this problem, position minimization is first made with a large step length $\sigma_{hi}$ approximated to that proposed by (Schwefel 1981) and after position optimization target is reached, orientation minimization is added to optimization with step length $\sigma_{low}$, that is ten times smaller. This strategy results in improved convergence time of the algorithm.

Rechenberg's success rule is used for controlling the size of $\sigma$, (Schwefel 1981). After every $N_b$ (number of design variables) iterations, the number of successes occurred over the preceding $10N_b$ mutations are revised. If this number is less than $2N_b$, step size is multiplied by a factor of 0.85, or divided by 0.85 if more than $2N_b$ successes occurred.

For the initial values of $\sigma$, Schwefel proposes to use the following estimation:

$$\sigma_i^0 = \frac{\Delta b_i}{\sqrt{N_b}} \tag{3}$$

where $\Delta b_i$ is the expected distance from the optimum for the corresponding design variable. Notwithstanding as the accuracy for this initial $\sigma$ value is not critical because the law of success seems to quickly adapt the step size, a generalized form is deduced to approximate initial Schwefel values

$$\sigma_i^0 = \frac{\Delta d_{q_N}}{10(N_b + 1)} \tag{4}$$

where $\Delta d_{q_N}$ is the last node distance error in millimeters. The value obtained here approximates experimental results average of Schwefel estimations that made no significant differences on convergence times with respect to that of the exact estimation. When optimizing orientation, step length in (4) is reduced by a factor of ten.

There is no accurate rule for determining an appropriate parent population size $\mu$. A good indicator is to have as many or a few times as many members as the number of design variables; parent population size used here is $N_b = 6$.

On the contrary, some theoretical studies have been realized on $(1, \lambda)$ strategies (Schwefel 1981), where $\lambda$ is the offspring population size, to estimate the optimal ratio between $\lambda/\mu$. It has been shown that this ratio depends on the objective function and increases with its complexity. A ratio $\lambda/\mu$ equal to 5 can be considered as a good starting point, therefore an offspring population size of $\lambda = 30$ is chosen here.

Total error is the sum of the position error at each path point $k = 2, \ldots, N$, and the orientation error in the last two nodes $k = (N-1), N$, with weights $W_1 = 0.5$ and $W_2 = 1$ respectively, attaches greater importance to the last point where the robot is meant to perform the manipulation. Thereby, the joint configuration path must be transformed into end-effector position and orientation coordinates through the robot manipulator kinematic model. Position and orientation errors, denoted as $E_P$ and $E_0$, are defined as (González et al 2009)

$$E_P(\Omega) = \frac{1}{2R_{max}} \sum_{k=2}^{N} |p_{l_k} - p_k| \tag{5}$$

and

$$E_O(\Omega) = \frac{1}{2\pi} \sum_{k=N-1}^{N} |\varphi_{l_k} - \varphi_k|, \tag{6}$$

where $((p_l), (\varphi_l))$ are the desired position and orientation coordinates calculated by $\Omega_l$ forward kinematics, and $R_{max}$ is the robot manipulator's maximum reach suggested by (Tabandeh et al 2006) as a normalization value. The resulting optimal joint path $\Omega*$ minimizes the total deviation with respect to $\Omega_l$, and the optimization problem is realised by the minimization of:

$$f(\Omega) = w_1 E_P(\Omega) + w_2 E_O(\Omega) + w_3 \zeta(\Omega). \quad (7)$$

subjetc to:

$$C = \{\Omega \mid g(\Omega) \leq 0 \wedge h(\Omega) \geq 0\},$$

where $g$ and $h$ are restrictions imposed by the mechanical joint limits of the robot manipulator, and $w_1$, $w_2$ and $w_3$ are weighting factors used according to task priorities.

## 3 Evolution Strategies Adaptation Algorithm

The path evolutionary adaptation is accomplished with an implementation of the ES method denominated in (Datoussaid et al 2002). The algorithm used to adapt the manipulation path is illustrated in Algorithm 1, where $P_g$, and $Po_g$ are parent and offspring populations respectively, in generation $g$, and $n_b$ is the number of design variables, which corresponds to the number of manipulator joints.

**Algorithm 1** Evolution Strategies

```
initialization Pg = 0
evaluation Pg
while termination criterion ≠ true do
    Po_g ← Evolutionary mutation
    evaluation Po_g
    P_{g+1} ← selection(Po_g ⋃ P_g)
    if optimal position = true then
        reduce σ
    end if
    if g module(10 n_b) = 0 then
        step length control
    end if
    g ← g + 1
end while
```

The $(\mu+\lambda)$-EE presented in (Datoussaid et al 2002) is used with some modifications to address the optimal path adaptation problem. A known initial manipulator configuration vector $\overrightarrow{q_1}$ is assumed, as well as a robot base location and orientation at learned path $p_l$.

Consider an initial population of $\mu$ parent individuals defined as in (1)

$$P_g = \{\Omega_{1,g}, \ldots, \Omega_{i,g}, \ldots, \Omega_{\mu,g}\},$$

where $\Omega_i$ represents a floating point vector with size $T = N.n$ and $g = 0, \ldots, g_{max}$ the generation number. In the scheme $(\mu+\lambda)$-*EE* the initialization process generates a population of $\mu$ random individuals distributed within the vector parameter bounds. If the initial location is unknown, then the use of an uniform distribution would be advisable to ensure the diversity of the population. Furthermore, if the initial joint configuration $\overrightarrow{q_1}$ is considered close enough to the learnt path initial node ($k = 1$), then we can intuitively assume that the optimal solution must be near the learned path $\Omega_l$. This first estimation is included in the initialization process $P_{g=0}$, as the learnt path perturbation with a Gaussian probability distribution at the configuration nodes $K = 2, \ldots, N$, reducing the convergence time. Therefore, the initialization process can be expressed as

$$\Omega_{i,g} = \begin{cases} \overrightarrow{\hat{q}_1}, & \text{if } k = 1, \\ \overrightarrow{q_{l_k}} + rand_G(0, \sigma^2), & \text{if } k = 2, \ldots, N \end{cases} \quad (8)$$

where $rand_G(0, \sigma)$ is a Gaussian distribution random number generator with zero mean and standard deviation $\sigma$, and $i = 1, \ldots, \mu$.

Once the population has been initialized, mutation is used to build a $\lambda$ size offspring population. For each offspring, a parent is randomly selected, and each of its design variables $b_i$ is mutated by adding a Gaussian random variable with zero mean and a standard deviation $\sigma$.

$$\Omega_{j,g} = \begin{cases} \overrightarrow{\hat{q}_1}, & \text{if } k = 1, \\ \overrightarrow{q_{l_k}} + rand_G(0, \sigma^2), & \text{if } k = 2, \ldots, N \end{cases} \quad (9)$$

Where $j = 1, \ldots, \lambda$, the step length $\sigma = \sigma_{hi}$ during the position optimization phase and $\sigma = \sigma_{low}$ during the orientation optimization phase.

The objective evaluation function is used to assign a cost value to each member from parent and offspring populations: $P_g$ and $Po_g$. The new parents $P_{g+1}$ are selected from both populations as $\mu$ individuals with the best fitness, i.e. individuals with the lowest cost function.

$$\Omega_{i,g+1} = \begin{cases} \Omega_{j,g}, \, if \; f(\Omega_{j,g}) \leq f(\Omega_{i,g}) \\ \Omega_{i,g}, \, otherwise \end{cases} \quad (10)$$

The manipulator forward kinematics defined by an homogeneous transformation matrix is calculated to obtain total cost function on (7). Homogeneous transform is a four by four elements matrix that contains end-effector location used for (5) and a rotation sub-matrix that is used to determine orientation error for (6), in this way reduced computational time is achieved by avoiding exact angles calculation.

Optimization loop begins by minimizing the position error until a fitness value is reached, and then orientation error is added to the cost function. During the next few iterations the total error is incremented due to the influence of newly added criterion, but then both criterion errors are gently improved as the generations evolve.

The only drawback in obtaining $E_O$ from within the homogeneous transform is that the calculated error in (6) is not directly proportional to the angle error since it is the result of mathematical functions applied to the end-effector orientation angles; therefore it can't be used as a termination criterion. This issue is overcome by checking angles after position fitness is reached. Termination criterion takes into account only the position error until a fitness value is reached, then exact angles error is evaluated, if orientation fitness is not reached the position fitness value is reduced and minimization process continues. As both errors evolve together, orientation fitness is found eventually.

To ensure generation of a feasible path, joint upper and lower limits need to be revised during optimization process. Joint limits are mechanical constraints that define the manipulator workspace, but also represent configuration values of reduced dexterity and hence should be avoided in the execution of the task. In the case of a boundary constraint violation, there are many solutions to replace values that have exceeded their limits, (Price et al 2005). Here a simple strategy is used, resetting the out-of-bound parameters with the exceeded bound value.

Finally mutation-selection process continues until convergence criterion is achieved or until the maximum number of generations is reached.

## 4 Simulations and Experiments

The proposed methodology is tested in a simulation environment with a non-redundant mobile manipulator robot denominated MANFRED-2 (Blanco et al 2005), which consists of a six degrees of freedom (n = 6) anthropomorphic arm mounted over a two degrees of freedom mobile base (n=2). This robot was built at the Carlos III University of Madrid. Figure 2 shows the MANFRED-2 robot in the implemented 3D simulation environment; our laboratory was modeled with elements such as doors and small tools to test grasping and manipulation tasks, simulations include body dynamics to increase realism and assure veracity.

The Denavit-Hartenberg parameters and joint limits for mobile manipulator MANFRED-2's robotic arm, are presented in Table 1.

**Table 1** MANFRED-2 robot Denavit-Hartenberg parameters and joint limits.

| **Art.** | $\alpha_j$ | $a_j(m)$ | $\theta_j$ | $d_j$ | $q_j^{ba}$ | $q_j^{al}$ |
|---|---|---|---|---|---|---|
| 1 | 90 | 0 | 0 | 0.25 | -90 | 90 |
| 2 | -90 | 0.4 | 0 | 0 | 0 | 180 |
| 3 | -90 | 0 | -90 | 0 | -90 | 90 |
| 4 | 90 | 0 | 0 | 0.35 | -90 | 90 |
| 5 | -90 | 0 | 0 | 0 | -90 | 90 |
| 6 | 0 | 0 | 0 | 0.25 | -90 | 90 |

The simulation environment software is used in order to obtain a convenient manipulation path $\Omega_l$, this is accomplished by actioning servomotors separately until a desired pose is found. When a desired pose is reached, our software enables us to save the robot's configuration and move to the next point; each pose is saved to form a manipulation path that can be then executed as a sequence of poses that lead to the the goal reaching point.

The $\Omega_l$ path describes our known task with $N = 6$ points, as shown in Figure 2. Forward kinematics is calculated using Denavit and Hartenberg (1955), obtaining the end-effectors position and orientation $\{(x_k, y_k, z_k), (\phi_k, \theta_k, \psi_k)\}$ for each point $k = 1, 2, \ldots, 6$.

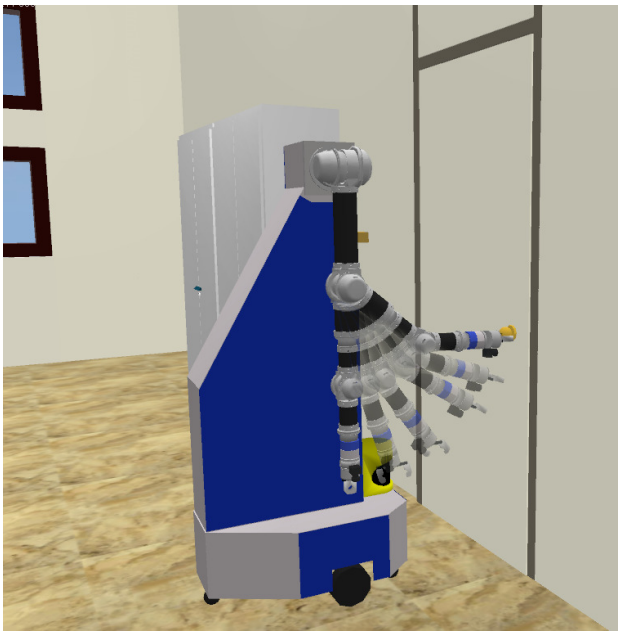

**Fig. 2** Manipulation learned path $\Omega_l$.

Table 2 shows the end-effector points coordinates for $\Omega_l$. The robot location is given by the parameters $(x_b, y_b, \theta_b)$ referenced to a point predefined on the simulation map, where $(x_b, y_b)$ determine the position coordinates in meters and $\theta_b$ the robot base orientation

in degrees, position in $z_b$ is not included since the robot base keeps the robotic arm at the same height all the time. For $\Omega_l$, the location is $p_l = (-2.319, -2.138, 180°)$.

Learned path $\Omega_l$ is tested on the 3D simulation environment, results show how the robot reaches an experimental tool. Subsequently, the door knob is grabbed when the robotic hand is closed, and robot is capable of opening the door by moving its base backwards.

**Table 2** End-effector learned path in Cartesian space.

| **k** | $X_k$ | $Y_k$ | $Z_k$ | $\phi_k$ | $\theta_k$ | $\psi_k$ |
|---|---|---|---|---|---|---|
| 1 | 250 | 147.63 | -1000 | $-180°$ | $0°$ | $-90°$ |
| 2 | 250 | 147.63 | -980.62 | $174.27°$ | $17.09°$ | $-108.86°$ |
| 3 | 250 | 284.83 | -923.95 | $156.88°$ | $28.39°$ | $-131.93°$ |
| 4 | 250 | 402.01 | -834.35 | $131.93°$ | $28.39°$ | $-156.88°$ |
| 5 | 250 | 491.23 | -718.62 | $108.86°$ | $17.09°$ | $-174.27°$ |
| 6 | 250 | 546.82 | -585.47 | $90°$ | $0°$ | $180°$ |

Two new random locations are used to verify the algorithm's effectiveness: $p_1$ and $p_2$, these locations are different in position and orientation from that of the known path. An initial robot arm configuration $\mathbf{q_1} = \{0°, 0°, 0°, 0°, 0°, 0°\}$ is assumed for all cases, and the offspring population size is $\lambda = 30$ as proposed in Section 2. The algorithm is executed 20 times for each location. Table 3 shows robot base locations for $\Omega_l$, $\Omega_1$ and $\Omega_2$.

**Table 3** Position and orientation coordinates of robot base for $\Omega_l$, $\Omega_1$ and $\Omega_2$

| Path | x(m) | y(m) | $\theta$ |
|---|---|---|---|
| $\Omega_l$ | -2.319 | -2.138 | $180.00°$ |
| $\Omega_1$ | -2.294 | -2.104 | $181.48°$ |
| $\Omega_2$ | -2.200 | -2.207 | $190.48°$ |

Once initial population members are generated using (8), the algorithm starts executing iteratively to minimize (7).

The mutation process of the candidate population is made via software, verifying that the generated poses are collision free, as evolving candidates in real robots is dangerous.

Position is optimized first with configuration parameters: $F = 0.012$ and $\sigma$ in accordance to (4). After fitness is reached, orientation is added to the objective function with $\sigma$ reduced by a factor of ten; termination criterion of end effector error to be less than 2.5 mm is being set. Test results are shown in Table 4.

For an execution of the algorithm at $p_1$, a solution $\Omega_1$ is found after $g = 120$ generations in 1.5 seconds. In orientation terms an error of $2.98°$ is obtained on the


**Fig. 3** Mobile Robot MANFRED-2 reaching a door knob.

$N-1$ point, and $0.29°$ on the last one. This makes sense when we recall the weighting factors for orientation optimization: $W_5 = 0.5$ and $W_6 = 1$. Lines described by the learned and evolutionary algorithm adapted path are shown on Figure 4, it can be seen that the adapted path fits position closely with a soften adaptation curve when approaching to the known path; a position error of 1.78 mm is obtained in last node for this test execution. Results showed that intermediate points presented lower position errors and greater orientation errors than others because they are only optimized in position, while last two nodes presented minimal orientation error.

All obtained manipulation paths are tested in the three-dimensional dynamic simulation environment as well as in the real robot where paths are executed and the sequences are reached correctly. The door is opened when additional steps are executed. The door knob had to be taped to increase the friction with the robot gripper and enable turning, but beyond this detail, the simulation represented precisely the real environment. Fig. 3 shows a picture of the robot reaching the door knob. The same perspective of Fig. **??** could not be presented because of a wall next to the robot's arm.

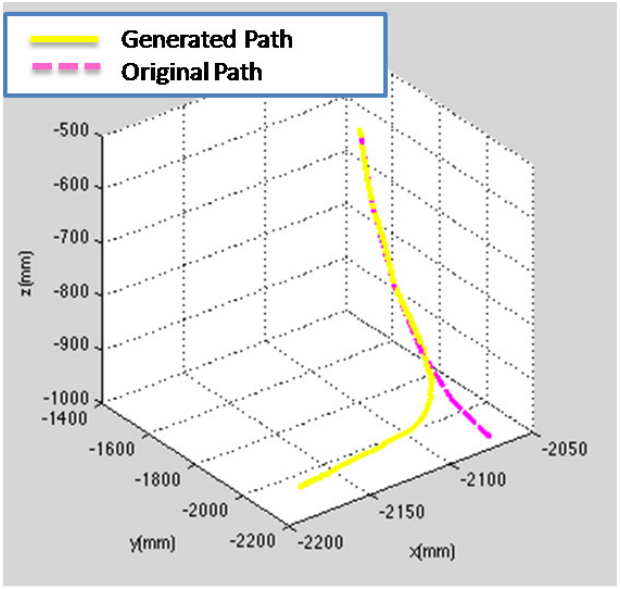


**Fig. 4** Adapted and learned manipulation paths, $\Omega_1$ and $\Omega_l$.

**Table 4** Path generarion statistics for $\Omega_1$ and $\Omega_2$.

| | Generated paths: | |
|---|---|---|
| | $\Omega_1$ | $\Omega_2$ |
| Number of tests | 20 | 20 |
| Mean convergence generation | 186.2 | 261.8 |
| Best convergence generation | 31 | 40 |
| Worst convergence generation | 500 | 500 |
| Mean convergence time | 2.18s | 2.71s |
| Best convergence time | 0.59s | 3.92s |
| Worst convergence time | 4.87s | 4.90s |
| Mean position error | 1.96mm | 1.73mm |
| Min. position error | 0.76mm | 0.28mm |
| Max. position error | 2.72mm | 1.22mm |
| Mean orientation error | 0.7831° | 0.3703° |
| Min. orientation error | 0.5679° | 0.0059° |
| Max. orientation error | 1.1180° | 0.8448° |

Experimental results show that errors can be minimized so the robot can carry out defined tasks. Time in worst-case scenario rose up to 4.9 seconds when reaching maximum generation $g_{max} = 500$, which stays within the proposed real-time threshold. Further investigation over our previous work Arismendi et al (2012) found that the forward kinematics calculus library consumed most of the algorithm computational time. Therefore, a more computationally optimal forward kinematics function was implemented reducing execution time significantly. When $g_{max}$ is reached the fitness distance between the best and the worst individuals in population is taken as the convergence criterion, observe closely Figure 5 and 6.

On account of a reduced parent population size $\mu = 6$, lines on Figures 5 and 6 followed closely. Also an error peak when orientation optimization begins at around generation 20 can be observed on Figure 6. This peak is less obvious on Figure 5 because robot base orientation error is smaller in that case.

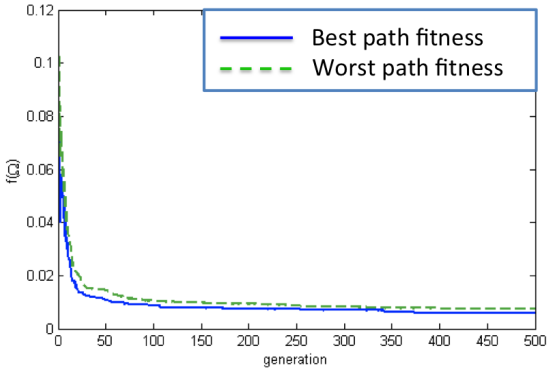


**Fig. 5** Path fitness evolution for $\Omega_1$ through $g_{max}$ generations.

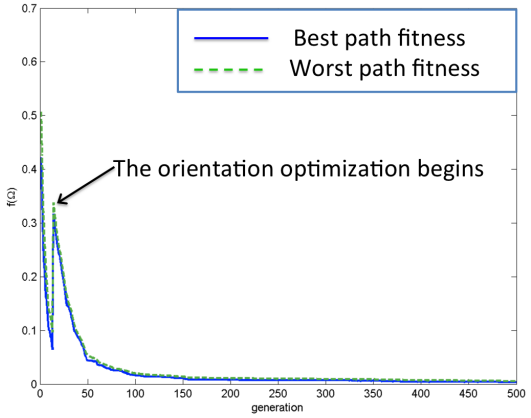


**Fig. 6** Path fitness evolution for $\Omega_2$ through $g_{max}$ generations.

## 5 Conclusions

A methodology built over ES has been presented. The adaptation of manipulation paths for mobile manipulators is possible in real-time, achieving optimal manipulation relative to position, orientation and energy consumption. Given a learned manipulation path a new one is calculated when robot base is in a different location from that of the learned path, minimal position and orientation end-effector errors are obtained within the

evolutionary process. Calculus are simplified to reduce computational time. A forward kinematics function was implemented for reaching optimal computational time, it implied a significant time reduction for the execution of the algorithm. Granted that the algorithm needs no inverse kinematics, singularities are avoided and convergence is guaranteed.

The experimental results showed the ability to apply the algorithm in real-time for obtaining adapted manipulation paths, proving to be a feasible solution for mobile robots manipulation problems. A computational time improvement was obtained by first optimizing position until position error is minimized, and then orientation error is added to the objective function with a reduced mutation step length until termination criterion is fulfilled at the end of the process. In addition, self-adjusting step length parameter was shown to perform efficiently compared to that of the DE algorithm. It is advisable to prioritize minimizing factors according to tasks because there is a proportional relationship between time and optimization parameters.

We are presently working on a number of extensions to our current work. First, the generation of paths with the specification of only one goal configuration, this seems to be easily implementable. Second, the use of efficient collision detection to avoid obstacles, this is going to be another optimization parameter, which on getting closer to obstacles will increase the error value.

Exploring these implementations, and conducting further analysis forms the basis of our future work.

**Acknowledgements** The research leading to these results has received funding from the RoboCity2030-II-CM project (S2009/DPI-1559), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds of the EU.

## References

Arismendi C, Gómez JV, Garrido S, Moreno L (2012) Adaptive Evolution Strategy for Robotic Manipulation. In: 2012 IEEE Conference on Evolving and Adaptive Intelligent Systems, IEEE, pp 29–34, DOI 10.1109/EAIS.2012.6232800

Bao-Liang L, Ito K (1995) Regularization of inverse kinematics for redundant manipulators using neural network inversions. In: Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE, pp 2726–2731, DOI 10.1109/ICNN.1995.488161

Bertram D, Kuffner J, Dillmann R, Asfour T (2006) An integrated approach to inverse kinematics and path planning for redundant manipulators. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., Ieee, pp 1874–1879, DOI 10.1109/ROBOT.2006.1641979

Blanco D, Ansari SA, Castejón C, López Boada B, Moreno LE (2005) MANFRED: Robot Antropomórfico De Servicio Fiable Y Seguro para operar En Entornos Humanos. Revista Iberoamericana de Ingeniería Mecánica 9(3):33–48

Borboni A (2001) Solution of the inverse kinematic problem of a serial manipulator by a fuzzy algorithm. IEEE, DOI 10.1109/FUZZ.2001.1007317

Chiaverini S, Egeland O, Kanestrom R (1991) Achieving user-defined accuracy with damped least-squares inverse kinematics. IEEE, DOI 10.1109/ICAR.1991.240676

Datoussaid S, Verlinden O, Conti C (2002) Application of Evolutionary Strategies to Optimal Design of Multibody Systems. Multibody System Dynamics 8(4):393–408, DOI 10.1023/A:1021101912826

Denavit J, Hartenberg R (1955) A kinematic notation for lower-pair mechanisms based on matrices. Journal of Applied Mechanics 23(June):215 – 221

Goldenberg A, Benhabib B, Fenton R (1985) A complete generalized solution to the inverse kinematics of robots. Robotics and Automation, IEEE Journal of 1(1):14–20

González C, Blanco D, Moreno L (2009) Optimum robot manipulator path generation using Differential Evolution. IEEE Congress on Evolutionary Computation, CEC, DOI 10.1109/CEC.2009.4983366

Guez A, Ahmad Z (1988) Solution to the inverse kinematics problem in robotics by neural networks. IEEE, DOI 10.1109/ICNN.1988.23979

Gupta K, Kazerounian K (1985) Improved numerical solutions of inverse kinematics of robots. Robotics and Automation Proceedings 1985 IEEE International Conference on pp 743–748

Huapeng W, Handroos H (2000) Utilization of differential evolution in inverse kinematics solution of a parallel redundant manipulator. In: KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516), IEEE, pp 812–815, DOI 10.1109/KES.2000.884170

Kim SW, Lee JJ, Sugisaka M (1993) Inverse kinematics solution based on fuzzy logic for redundant manipulators. IEEE, DOI 10.1109/IROS.1993.583249

Kumbla K, Jamshidi M (1994) Control of robotic manipulator using fuzzy logic. IEEE, DOI 10.1109/FUZZY.1994.343731

Laplante PA (2004) Real-time systems design and analysis, third edit edn. Wiley-IEEE, United States of America

Mayorga R, Wong A, Milano N (1992) A Fast Damped Least-squares Solution To Manipulator Inverse Kinematics And Singularities Prevention. Intelligent Robots and Systems, 1992, Proceedings of the 1992 IEEE/RSJ International Conference on pp 1177–1184

Morris AS, Mansor A (1997) Finding the inverse kinematics of manipulator arm using artificial neural network with lookup table. Robotica 15(May 2010):617 – 625

Oyama E, Agah A, Maeda T (2001) Inverse kinematics learning by modular architecture neural networks with performance prediction networks. IEEE, DOI 10.1109/ROBOT.2001.932681

Parker J, Khoogar A, Goldberg D (1989) Inverse kinematics of redundant robots using genetic algorithms. IEEE Comput. Soc. Press, DOI 10.1109/ROBOT.1989.100000

Price K, Storn RM, Lampinen JA (2005) Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series). Springer

Schwefel HP (1981) Numerical Optimization of Computer Models. John Wiley & Sons Ltd

Shen W, Gu J, Milios EE (2006) Self-Configuration Fuzzy System for Inverse Kinematics of Robot Manipulators. IEEE, DOI 10.1109/NAFIPS.2006.365856

Tabandeh S, Clark C, Melek W (2006) A Genetic Algorithm Approach to solve for Multiple Solutions of Inverse Kinematics using Adaptive Niching and Clustering. Evolutionary Computation, 2006 CEC 2006 IEEE Congress on

Tejomurtula S, Kak S (1999) Inverse kinematics in robotics using neural networks. Information Sciences 116(2-4):147–164, DOI 10.1016/S0020-0255(98)10098-1

Thiang H, Pangaldus R (2009) Artificial Neural Network with Steepest Descent Backpropagation Training Algorithm for Modeling Inverse Kinematics of Manipulator. World Academy of Science, Engineering and Technology pp 671–674

Vande Weghe M, Ferguson D, Srinivasa SS (2007) Randomized path planning for redundant manipulators without inverse kinematics. 2007 7th IEEE-RAS International Conference on Humanoid Robots pp 477–482, DOI 10.1109/ICHR.2007.4813913

Wang LC, Chen C (1991) A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. IEEE Transactions on Robotics and Automation 7(4):489–499, DOI 10.1109/70.86079

Xu H, Collins JJ (2009) Estimating the Odometry Error of a Mobile Robot by Neural Networks. IEEE, DOI 10.1109/ICMLA.2009.96

Yang M, Lu G, Li J (2001) An inverse kinematics solution for manipulators based on fuzzy logic. IEEE, DOI 10.1109/ICII.2001.983851